PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/38 — A1

(11) International Publication Number: **WO 98/20416**

(43) International Publication Date: 14 May 1998 (14.05.98)

(21) International Application Number: PCT/US96/17516

(22) International Filing Date: 4 November 1996 (04.11.96)

(71) Applicant *(for all designated States except US)*: ADVANCED MICRO DEVICES, INC. [US/US]; One AMD Place, Sunnyvale, CA 94088 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: PICKETT, James, K. [-/US]; 9802 Scenic Bluff Drive, Austin, TX 78733 (US).

(74) Agent: KIVLIN, B., Noel; Conley, Rose & Tayon, P.C., P.O. Box 3267, Houston, TX 77253-3267 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

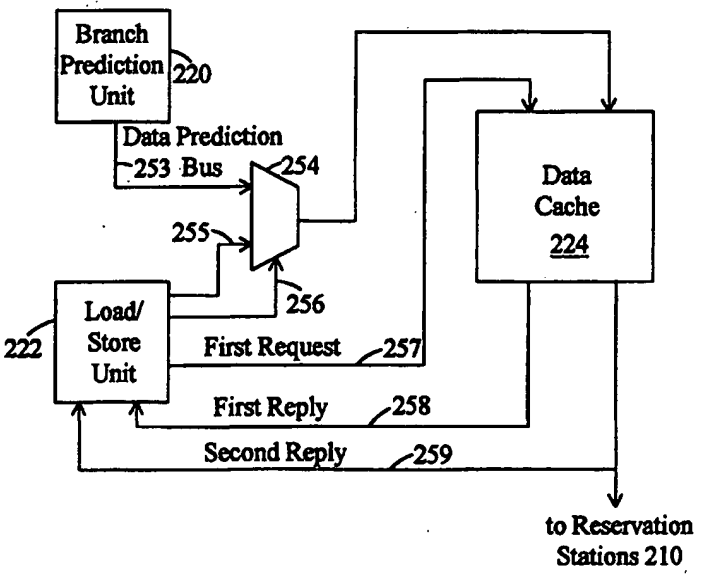
Published

With international search report.

(54) Title: A STRIDE-BASED DATA ADDRESS PREDICTION STRUCTURE

(57) Abstract

A data prediction structure is provided for a superscalar microprocessor. The data prediction structure stores base addresses and stride values in a prediction array. The base address and the stride value from a location within the data prediction structure indexed by an instruction address are added to form a data prediction address which is then used to fetch data bytes into a reservation station storing an associated instruction. If the data associated with an operand address calculated by an associated functional unit resides in the reservation station, the clock cycles use to perform the load operation have occurred before the instruction reached the reservation station. Additionally, the base address is updated to the address generated by executing an instruction each time the instruction is executed, and the stride value is updated when the data prediction address is found to be incorrect.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

TITLE: A STRIDE-BASED DATA ADDRESS PREDICTION STRUCTURE

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of superscalar microprocessors and, more particularly, to data prediction mechanisms in superscalar microprocessors.

2. Description of the Relevant Art

Superscalar microprocessors achieve high performance by simultaneously executing multiple instructions in a clock cycle and by specifying the shortest possible clock cycle consistent with the design. As used herein, the term "clock cycle" refers to an interval of time during which the pipeline stages of a microprocessor perform their intended functions. For example, superscalar microprocessors are typically configured with instruction processing pipelines which process instructions. The processing of instructions includes the actions of fetching, dispatching, decoding, executing, and writing back results. Each action may be implemented in one or more pipeline stages, and an instruction flows through each of the pipeline stages where an action or portion of an action is performed. At the end of a clock cycle, the instruction and the values resulting from performing the action of the current pipeline stage are moved to the next pipeline stage. When an instruction reaches the end of an instruction processing pipeline, it is processed and the results of executing the instruction have been recorded.

Since superscalar microprocessors execute multiple instructions per clock cycle and the clock cycle is short, a high bandwidth memory system is required to provide instructions and data to the superscalar microprocessor (i.e. a memory system that can provide a large number of bytes in a short period of time). Without a high bandwidth memory system, the microprocessor would spend a large number of clock cycles waiting for instructions or data to be provided, then would execute the received instructions and/or instructions dependent upon the received data in a relatively small number of clock cycles. Overall performance would be degraded by the large number of idle clock cycles. Superscalar microprocessors are, however, ordinarily configured into computer systems with a relatively large main memory composed of dynamic random access memory (DRAM) cells. DRAM cells are characterized by access times which are significantly longer than the clock cycle of modern superscalar microprocessors. Also, DRAM cells typically provide a relatively narrow output bus to convey the stored bytes to the superscalar microprocessor. Therefore, DRAM cells provide a memory system that provides a relatively small number of bytes in a relatively long period of time, and do not form a high bandwidth memory system.

Because superscalar microprocessors are typically not configured into a computer system with a memory system having sufficient bandwidth to continuously provide instructions and data, superscalar

microprocessors are often configured with caches. Caches include multiple blocks of storage locations configured on the same silicon substrate as the microprocessor or coupled nearby. The blocks of storage locations are used to hold previously fetched instruction or data bytes. The bytes can be transferred from the cache to the destination (a register or an instruction processing pipeline) quickly; commonly one or two clock cycles are required as opposed to a large number of clock cycles to transfer bytes from a DRAM main memory.

Unfortunately, the latency of one or two clock cycles within a data cache is becoming a performance problem for superscalar microprocessors as they attempt to execute more instructions per clock cycle. The problem is of particular importance to superscalar microprocessors configured to execute instructions from a complex instruction set architecture such as the x86 architecture. As will be appreciated by one skilled in the art, x86 instructions allow for one of their "operands" (the values that the instruction operates on) to be stored in a memory location. Instructions which have an operand stored in memory are said to have an implicit (memory read) operation and an explicit operation which is defined by the particular instruction being executed (i.e. an add instruction has an explicit operation of addition). Such an instruction therefore requires an implicit address calculation for the memory read to retrieve the operand, the implicit memory read, and the execution of the explicit operation of the instruction (for example, an addition). Typical superscalar microprocessors have required that these operations be performed by the execute stage of the instruction processing pipeline. The execute stage of the instruction processing pipeline is therefore occupied for several clock cycles when executing such an instruction. For example, a superscalar microprocessor might require one clock cycle for the address calculation, one to two clock cycles for the data cache access, and one clock cycle for the execution of the explicit operation of the instruction. Conversely, instructions with operands stored in registers configured within the microprocessor retrieve the operands before they enter the execute stage of the instruction processing pipeline, since no address calculation is needed to locate the operands. Instructions with operands stored in registers would therefore only require the one clock cycle for execution of the explicit operation. A structure allowing the retrieval of memory operands for instructions before they enter the execute stage is therefore desired.

SUMMARY OF THE INVENTION

The problems outlined above are in large part solved by a data prediction structure for a superscalar microprocessor in accordance with the present invention. The data prediction structure stores base addresses and stride values in a prediction array. A particular base address and associated stride value are added to form a data prediction address. The data prediction address is then used to fetch data bytes which are conveyed to the reservation stations of the superscalar microprocessor. If the data associated with an operand address calculated by a functional unit resides in the reservation station, the data is used as the operand. Advantageously, the clock cycles used to perform the load operation occur before the instruction reaches the reservation station. The instruction occupies the reservation station for fewer clock cycles than would be necessary utilizing the conventional method of executing these instructions. The clock cycles that

are saved may be profitably used to store other instructions.

Additionally, the base address is updated to the address generated by a functional unit each time an associated instruction is executed, and the stride value is updated when the data prediction address is found to be incorrect. In this way, the data prediction address is in many cases more accurate then a static data prediction address that changes only when it is found to be incorrect. Advantageously, the implicit load may be performed more often prior to the instruction reaching the reservation station. Furthermore, if several correct predictions are made in consecutive executions of an associated instruction, the prediction must be incorrect in several consecutive executions before the stride is changed (according to one embodiment of the present invention). Advantageously, a single execution of another instruction whose instruction address indexes the same storage location as the correct prediction information does not destroy the prediction information.

Broadly speaking, the present invention contemplates a method for predicting a data address which will be referenced by a plurality of instructions when said plurality of instructions are fetched, comprising several steps. A data prediction address is generated from a base address and a stride value during a clock cycle in which a data prediction counter indicates that the base address and the stride value are valid. Data associated with the data prediction is fetched from a data cache. Then, the data is stored within a plurality of reservation stations.

The present invention further contemplates a data address prediction structure comprising an array, an adder circuit, and a reservation station. The array includes a plurality of storage locations for storing a plurality of base addresses and a plurality of stride values. A particular storage location is selected by an instruction address which identifies an instruction. The instruction generates one of the plurality of base addresses. The adder circuit is coupled to the array for adding a base address and a stride value conveyed from the array to produce a data prediction address. The reservation station stores the instruction and data associated with the data prediction address.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects and advantages of the invention will become apparent upon reading the following detailed description and upon reference to the accompanying drawings in which:

Figure 1 is a block diagram of a superscalar microprocessor.

Figure 2 is a diagram depicting a branch prediction unit, a load/store unit, and a data cache of the superscalar microprocessor of Figure 1, along with several elements of one embodiment of the present invention.

Figure 3A is a diagram of the branch prediction unit shown in Figure 2 depicting several elements of one embodiment of the present invention.

Figure 3B is a diagram of a reservation station of the superscalar microprocessor shown in Figure 1 showing several elements of one embodiment of the present invention.

Figure 4 is a diagram of the information stored in a storage location of a branch prediction array in accordance with the present invention.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF THE INVENTION

Turning now to Figure 1, a block diagram of a superscalar microprocessor 200 including reservation stations 210 and a branch prediction unit 220 in accordance with the present invention is shown. As illustrated in the embodiment of Figure 1, superscalar microprocessor 200 includes a prefetch/predecode unit 202 and a branch prediction unit 220 coupled to an instruction cache 204. Instruction alignment unit 206 is coupled between instruction cache 204 and a plurality of decode units 208A-208F (referred to collectively as decode units 208). Each decode unit 208A-208F is coupled to respective reservation station units 210A-210F (referred to collectively as reservation stations 210), and each reservation station 210A-210F is coupled to a respective functional unit 212A-212F (referred to collectively as functional units 212). Decode units 208, reservation stations 210, and functional units 212 are further coupled to a reorder buffer 216, a register file 218 and a load/store unit 222. A data cache 224 is finally shown coupled to load/store unit 222, and an MROM unit 209 is shown coupled to instruction alignment unit 206.

Generally speaking, instruction cache 204 is a high speed cache memory provided to temporarily store instructions prior to their dispatch to decode units 208. In one embodiment, instruction cache 204 is configured to cache up to 32 kilobytes of instruction code organized in lines of 16 bytes each (where each byte consists of 8 bits). During operation, instruction code is provided to instruction cache 204 by prefetching code from a main memory (not shown) through prefetch/predecode unit 202. It is noted that instruction cache 204 could be implemented in a set-associative, a fully-associative, or a direct-mapped configuration.

Prefetch/predecode unit 202 is provided to prefetch instruction code from the main memory for storage within instruction cache 204. In one embodiment, prefetch/predecode unit 202 is configured to burst 64-bit wide code from the main memory into instruction cache 204. It is understood that a variety of specific code prefetching techniques and algorithms may be employed by prefetch/predecode unit 202.

As prefetch/predecode unit 202 fetches instructions from the main memory, it generates three predecode bits associated with each byte of instruction code: a start bit, an end bit, and a "functional" bit. The predecode bits form tags indicative of the boundaries of each instruction. The predecode tags may also convey additional information such as whether a given instruction can be decoded directly by decode units 208 or whether the instruction must be executed by invoking a microcode procedure controlled by MROM unit 209, as will be described in greater detail below.

Table 1 indicates one encoding of the predecode tags. As indicated within the table, if a given byte is the first byte of an instruction, the start bit for that byte is set. If the byte is the last byte of an instruction, the end bit for that byte is set. If a particular instruction cannot be directly decoded by the decode units 208, the functional bit associated with the first byte of the instruction is set. On the other hand, if the instruction can be directly decoded by the decode units 208, the functional bit associated with the first byte of the instruction is cleared. The functional bit for the second byte of a particular instruction is cleared if the opcode is the first byte, and is set if the opcode is the second byte. It is noted that in situations where the opcode is the second byte, the first byte is a prefix byte. The functional bit values for instruction byte numbers 3-8 indicate whether the byte is a MODRM or an SIB byte, or whether the byte contains displacement or immediate data.

Table 1. Encoding of Start, End and Functional Bits

Instr. Byte Number	Start Bit Value	End Bit Value	Functional Bit Value	Meaning
1	1	X	0	Fast decode
1	1	X	1	MROM instr.
2	0	X	0	Opcode is first byte
2	0	X	1	Opcode is this byte, first byte is prefix
3-8	0	X	0	Mod R/M or SIB byte
3-8	0	X	1	Displacement or immediate data; the second functional bit set in bytes 3-8 indicates immediate data
1-8	X	0	X	Not last byte of instruction
1-8	X	1	X	Last byte of instruction

As stated previously, in one embodiment certain instructions within the x86 instruction set may be directly decoded by decode units 208. These instructions are referred to as "fast path" instructions. The

remaining instructions of the x86 instruction set are referred to as "MROM instructions". MROM instructions are executed by invoking MROM unit 209. More specifically, when an MROM instruction is encountered, MROM unit 209 parses and serializes the instruction into a subset of defined fast path instructions to effectuate a desired operation. A listing of exemplary x86 instructions categorized as fast path instructions as well as a description of the manner of handling both fast path and MROM instructions will be provided further below.

Instruction alignment unit 206 is provided to channel variable byte length instructions from instruction cache 204 to fixed issue positions formed by decode units 208A-208F. Instruction alignment unit 206 independently and in parallel selects instructions from three groups of instruction bytes provided by instruction cache 204 and arranges these bytes into three groups of preliminary issue positions. Each group of issue positions is associated with one of the three groups of instruction bytes. The preliminary issue positions are then merged together to form the final issue positions, each of which is coupled to one of decode units 208.

Before proceeding with a detailed description of the data address prediction structure embodied in reservation stations 210 and branch prediction unit 220, general aspects regarding other subsystems employed within the exemplary superscalar microprocessor 200 of Figure 1 will be described. For the embodiment of Figure 1, each of the decode units 208 includes decoding circuitry for decoding the predetermined fast path instructions referred to above. In addition, each decode unit 208A-208F routes displacement and immediate data to a corresponding reservation station unit 210A-210F. Output signals from the decode units 208 include bit-encoded execution instructions for the functional units 212 as well as operand address information, immediate data and/or displacement data.

The superscalar microprocessor of Figure 1 supports out of order execution, and thus employs reorder buffer 216 to keep track of the original program sequence for register read and write operations, to implement register renaming, to allow for speculative instruction execution and branch misprediction recovery, and to facilitate precise exceptions. As will be appreciated by those of skill in the art, a temporary storage location within reorder buffer 216 is reserved upon decode of an instruction that involves the update of a register to thereby store speculative register states. Reorder buffer 216 may be implemented in a first-in-first-out configuration wherein speculative results move to the "bottom" of the buffer as they are validated and written to the register file, thus making room for new entries at the "top" of the buffer. Other specific configurations of reorder buffer 216 are also possible, as will be described further below. If a branch prediction is incorrect, the results of speculatively-executed instructions along the mispredicted path can be invalidated in the buffer before they are written to register file 218.

The bit-encoded execution instructions and immediate data provided at the outputs of decode units 208A-208F are routed directly to respective reservation station units 210A-210F. In one embodiment, each reservation station unit 210A-210F is capable of holding instruction information (i.e., bit encoded execution bits as well as operand values, operand tags and/or immediate data) for up to three pending instructions

awaiting issue to the corresponding functional unit. It is noted that for the embodiment of Figure 1, each decode unit 208A-208F is associated with a dedicated reservation station unit 210A-210F, and that each reservation station unit 210A-210F is similarly associated with a dedicated functional unit 212A-212F. Accordingly, six dedicated "issue positions" are formed by decode units 208, reservation station units 210 and functional units 212. Instructions aligned and dispatched to issue position 0 through decode unit 208A are passed to reservation station unit 210A and subsequently to functional unit 212A for execution. Similarly, instructions aligned and dispatched to decode unit 208B are passed to reservation station unit 210B and into functional unit 212B, and so on.

Upon decode of a particular instruction, if a required operand is a register location, register address information is routed to reorder buffer 216 and register file 218 simultaneously. Those of skill in the art will appreciate that the x86 register file includes eight 32 bit real registers (i.e., typically referred to as EAX, EBX, ECX, EDX, EBP, ESI, EDI and ESP). Reorder buffer 216 contains temporary storage locations for results which change the contents of these registers to thereby allow out of order execution. A temporary storage location of reorder buffer 216 is reserved for each instruction which, upon decode, is determined to modify the contents of one of the real registers. Therefore, at various points during execution of a particular program, reorder buffer 216 may have one or more locations which contain the speculatively executed contents of a given register. If following decode of a given instruction it is determined that reorder buffer 216 has a previous location or locations assigned to a register used as an operand in the given instruction, the reorder buffer 216 forwards to the corresponding reservation station either: 1) the value in the most recently assigned location, or 2) a tag for the most recently assigned location if the value has not yet been produced by the functional unit that will eventually execute the previous instruction. If the reorder buffer has a location reserved for a given register, the operand value (or tag) is provided from reorder buffer 216 rather than from register file 218. If there is no location reserved for a required register in reorder buffer 216, the value is taken directly from register file 218. If the operand corresponds to a memory location, the operand value is provided to the reservation station unit through load/store unit 222.

Details regarding suitable reorder buffer implementations may be found within the publication "Superscalar Microprocessor Design" by Mike Johnson, Prentice-Hall, Englewood Cliffs, New Jersey, 1991, and within the co-pending, commonly assigned patent application entitled "High Performance Superscalar Microprocessor", Serial No. 08/146,382, filed October 29, 1993 by Witt, et al. These documents are incorporated herein by reference in their entirety.

Reservation station units 210A-210F are provided to temporarily store instruction information to be speculatively executed by the corresponding functional units 212A-212F. As stated previously, each reservation station unit 210A-210F may store instruction information for up to three pending instructions. Each of the six reservation stations 210A-210F contain locations to store bit-encoded execution instructions to be speculatively executed by the corresponding functional unit and the values of operands. If a particular operand is not available, a tag for that operand is provided from reorder buffer 216 and is stored within the corresponding reservation station until the result has been generated (i.e., by completion of the execution of a

previous instruction). It is noted that when an instruction is executed by one of the functional units 212A-212F, the result of that instruction is passed directly to any reservation station units 210A-210F that are waiting for that result at the same time the result is passed to update reorder buffer 216 (this technique is commonly referred to as "result forwarding"). Instructions are issued to functional units for execution after the values of any required operand(s) are made available. That is, if an operand associated with a pending instruction within one of the reservation station units 210A-210F has been tagged with a location of a previous result value within reorder buffer 216 which corresponds to an instruction which modifies the required operand, the instruction is not issued to the corresponding functional unit 212 until the operand result for the previous instruction has been obtained. Accordingly, the order in which instructions are executed may not be the same as the order of the original program instruction sequence. Reorder buffer 216 ensures that data coherency is maintained in situations where read-after-write dependencies occur. Reservation stations 210 additionally store information related to the data prediction structure disclosed herein, as will be described below.

In one embodiment, each of the functional units 212 is configured to perform integer arithmetic operations of addition and subtraction, as well as shifts, rotates, logical operations, and branch operations. It is noted that a floating point unit (not shown) may also be employed to accommodate floating point operations.

Each of the functional units 212 also provides information regarding the execution of conditional branch instructions to the branch prediction unit 220. If a branch prediction was incorrect, branch prediction unit 220 flushes instructions subsequent to the mispredicted branch that have entered the instruction processing pipeline, and causes prefetch/predecode unit 202 to fetch the required instructions from instruction cache 204 or main memory. It is noted that in such situations, results of instructions in the original program sequence which occur after the mispredicted branch instruction are discarded, including those which were speculatively executed and temporarily stored in load/store unit 222 and reorder buffer 216. Exemplary configurations of suitable branch prediction mechanisms are well known.

Results produced by functional units 212 are sent to the reorder buffer 216 if a register value is being updated, and to the load/store unit 222 if the contents of a memory location is changed. If the result is to be stored in a register, the reorder buffer 216 stores the result in the location reserved for the value of the register when the instruction was decoded. As stated previously, results are also broadcast to reservation station units 210A-210F where pending instructions may be waiting for the results of previous instruction executions to obtain the required operand values.

Generally speaking, load/store unit 222 provides an interface between functional units 212A-212F and data cache 224. In one embodiment, load/store unit 222 is configured with a load/store buffer with eight storage locations for data and address information for pending loads or stores. Decode units 208 arbitrate for access to the load/store unit 222. When the buffer is full, a decode unit must wait until the load/store unit 222 has room for the pending load or store request information. The load/store unit 222 also performs

dependency checking for load instructions against pending store instructions to ensure that data coherency is maintained.

Data cache 224 is a high speed cache memory provided to temporarily store data being transferred between load/store unit 222 and the main memory subsystem. In one embodiment, data cache 224 has a capacity of storing up to eight kilobytes of data. It is understood that data cache 224 may be implemented in a variety of specific memory configurations, including a set associative configuration.

Turning now to Figure 2, several elements of one embodiment of a data address prediction structure are shown. Branch prediction unit 220 is included, along with load/store unit 222 and data cache 224. Branch prediction unit 220 is connected to a data prediction bus 253 which is coupled to an arbitration multiplexor 254. Also coupled to arbitration multiplexor 254 is a second request bus 255 and an arbitration select line 256. In this embodiment, signals on both second request bus 255 and arbitration select line 256 originate in load/store unit 222. The output bus of arbitration multiplexor 254 is coupled to an input port of data cache 224. A first request bus 257 is coupled between load/store unit 222 and an input port of data cache 224. Data cache 224 is configured with two output ports which are coupled to a first reply bus 258 and a second reply bus 259. Both first reply bus 258 and second reply bus 259 are coupled to load/store unit 222, and second reply bus 259 is coupled to reservation stations 210 (shown in Figure 1).

Generally speaking, branch prediction unit 220 produces a data prediction address on data prediction bus 253 during a clock cycle. The data prediction address is a prediction of the data addresses that are used by instructions being fetched during the clock cycle, and is generated from a stored set of addresses and associated stride values as will be detailed below.

During clock cycles in which the data prediction address is selected by arbitration multiplexor 254, the data prediction address accesses data cache 224. The data bytes associated with the data prediction address are transferred on second reply bus 259 to reservation stations 210 if the data prediction address is a hit in data cache 224. Reservation stations 210 store the data bytes along with the data prediction address and the associated instructions provided by decode units 208. Reservation stations 210, in a later clock cycle, direct the corresponding functional units 212 to generate a linear address if one or more of the associated instructions have an implicit memory read operation. If the linear address matches the data prediction address, than the data stored in the respective reservation station 210 is used as the operand for the instruction. Load/store unit 222 is informed by the corresponding reservation station 210A-210F that the associated implicit memory read operation need not be performed. Load/store unit 222 thereby discards the implicit memory read operation transferred by decode units 208 when the associated instruction was decoded. Therefore, the implicit memory read associated with an instruction may be performed before the instruction enters a functional unit 212. The latency of data cache 224 is endured prior to the instruction reaching the reservation station. Performance may be advantageously increased by allowing subsequent instructions to more quickly enter reservation stations 210 during clock cycles in which the associated instruction would have occupied reservation stations 210 due to the latency of data cache 224.

In another embodiment, load/store unit 222 stores data associated with data prediction addresses. When a functional unit 212 computes a linear address and it matches data stored by load/store unit 222, then load/store unit 222 conveys the stored data to the respective reservation station 210 without accessing data cache 224. Advantageously, data is provided for a load memory access without enduring the latency associated with data cache 224.

When a data prediction address misprediction is detected (as described below with respect to Figures 3A and 3B), the instruction address associated with the data prediction address and the correct data prediction address are conveyed to branch prediction unit 220. Branch prediction unit 220 stores the correct prediction address for use in a subsequent prediction.

The data prediction address is conveyed on data prediction bus 253 to arbitration multiplexor 254, which arbitrates between the data prediction address and a second request from load/store unit 222. The term arbitration, as used herein, means the selection of one request over another according to an arbitration scheme. In one embodiment, the arbitration scheme is a priority scheme in which a load/store request conveyed on second request bus 255 is a higher priority that the data prediction address request. Therefore, the arbitration select signal conveyed on arbitration select line 256 is an indication that a load/store request is being conveyed on second request bus 255. If a valid load/store request is being made during a clock cycle, arbitration multiplexor 254 selects the load/store request to be conveyed to data cache 224. If no valid load/store request is being made during a clock cycle, arbitration multiplexor 254 selects the data prediction address to be conveyed to data cache 224.

In one embodiment, load/store unit 222 is configured to make up to two requests to data cache 224 during a clock cycle. During clock cycles in which load/store unit 222 is making one request, first request bus 257 is used to allow the data prediction address the maximum opportunity to access data cache 224. If a data prediction address request is valid during a clock cycle in which load/store unit 222 is making no requests or one request to data cache 224, then the data prediction address request will be given access to data cache 224. Load/store unit 222 receives the data bytes associated with requests on first request bus 257 and second request bus 255 on first reply bus 258 and second reply bus 259, respectively. It is noted that other embodiments of load/store unit 222 may have different numbers of request buses for data cache 224.

Turning now to Figure 3A, an embodiment of branch prediction unit 220 is shown. The data prediction information is stored within a prediction array 270. In this embodiment, prediction array 270 is a linear array of storage locations. The current fetch address (conveyed from instruction cache 204) is used to index into prediction array 270 and select a storage location. Stored within the selected storage location is a base address, a stride value, and a data prediction counter. The base address is a data address generated by an instruction, wherein the address indicative of the memory location storing the instruction indexes to the selected storage location. The stride value is the difference between two addresses generated by the instruction on two consecutive executions of the instruction. The stride value is a signed value allowing both

positive and negative strides to be stored. The data prediction counter is configured to indicate the validity of the base address and the stride value. Furthermore, the data prediction counter may store values indicating that a number of previous data prediction addresses were correct predictions. A comparable number of consecutive mispredictions must then be detected before the stride value is changed. Generally speaking, the data prediction counter value may be incremented and decremented but does not decrement below zero nor increment above the largest value that may be represented by the data prediction counter. Instead, if the data prediction counter contains zero and is decremented, it remains zero. If the data prediction counter contains the largest value that it may represent and is incremented, it remains at the largest value that it may represent. The data prediction counter is therefore a saturating counter.

An adder circuit 271 is coupled to the output port of prediction array 270. Adder circuit 271 is configured to add the stride value stored within the selected storage location to the base address stored within the selected storage location, creating a data prediction address. The data prediction address is conveyed on data prediction bus 253, along with the associated data prediction counter value. In one embodiment, if the stride value is invalid (as indicated by the data prediction counter value), then the base address is added to zero.

In one embodiment, branch prediction unit 220 receives four sets of inputs from reservation stations 210. First, branch prediction unit 220 receives a plurality of linear address buses 261 which convey the actual data addresses (referred to below as linear addresses) generated by functional units 212 as well as a valid indicator for each address indicating the validity of the address. If a reservation station 210A-210F detects a mispredicted data prediction address, the associated one of linear address buses 261 will convey the corrected addresses (as will be explained below with respect to Figure 3B). Each reservation station 210A-210F also provides one of a plurality of mispredicted lines 273 indicating whether or not the linear address conveyed on the associated one of linear address buses 261 matches the data prediction address provided with the associated instruction. Additionally, branch prediction unit 220 receives a plurality of instruction address buses 274 from reservation stations 210 conveying instruction addresses associated with the linear addresses received on linear address buses 261. The instruction address is used to index into prediction array 270 in order to store the updated base address, stride value, and data prediction counter value. Finally, prediction address buses 276 are received by branch prediction unit 220. Prediction address buses 276 convey the data prediction address received by each reservation station 210A-210F along with the associated data prediction counter value. Linear address buses 261, mispredicted lines 273, instruction address buses 274 and prediction address buses 276 are received by a prediction validation and correction block 275 within branch prediction unit 220.

If one of mispredicted lines 273 signals an incorrect data address prediction and the corresponding address valid indicator from the associated one of linear address buses 261 indicates the linear address is valid, then a data address misprediction has occurred. Prediction validation and correction block 275 causes the corresponding linear address to be stored as the base address in prediction array 270 in the storage location indexed by the corresponding instruction address. The associated data prediction counter value is

decremented for this case and stored as the data prediction counter in the indexed storage location. In this manner, a data prediction address is corrected if predicted incorrectly. Additionally, a base address is created if a valid base address was not previously stored in prediction array 270 as indicated by the associated data prediction counter. A particular one of mispredicted lines 273 indicates a misprediction if the data prediction address associated with the instruction currently being validated is invalid. Therefore, the address provided on the associated one of linear address buses 261 is stored as the base address.

If a particular one of mispredicted lines 273 signals a correct prediction and the corresponding address valid indicator conveyed on the corresponding one of linear address buses 261 is indicative of a valid linear address, then the data prediction address is correctly predicted. Prediction validation and correction block 275 causes the corresponding linear address to be stored as the base address within prediction array 270 in the storage location indexed by the corresponding branch instruction address. The corresponding data prediction counter value is incremented and stored as the data prediction counter in the indexed storage location.

If the address valid indicator of a particular one of linear address buses 261 is not indicative of a valid linear address, then no prediction validation information is conveyed with respect to the instructions stored by the corresponding reservation station.

In either the misprediction case or correct prediction case detailed above, another check is performed. If the current data prediction counter value (i.e. prior to incrementing or decrementing) indicates that the stride value is invalid but the base address is valid, a new stride value is calculated by prediction validation and correction block 275. The new stride value is the difference between the prediction value conveyed on the associated one of prediction address buses 276 and the linear address conveyed on the associated one of linear address buses 261. The new stride value is stored as the stride value in the indexed storage location within prediction array 270. If the stride value is invalid (as indicated by the data prediction counter) then the data prediction counter is incremented regardless of the misprediction/prediction status of the current data address prediction.

For mispredicted lines and address valid indicators in a given clock cycle, reservation station 210A's outputs are given highest priority, then reservation station 210B's outputs, etc. In other words, if reservation station 210A is conveying prediction validation information during a clock cycle (as indicated by the associated address valid indicator), then reservation station 210A's information is processed by prediction validation and correction block 275. If reservation station 210A is not conveying prediction validation information during a clock cycle and reservation station 210B is conveying prediction validation information, then reservation station 210B's information is processed, etc. Prediction validation and correction block 275 is coupled to a dedicated write port into prediction array 270 to perform its updates, in one embodiment.

Because the prediction is formed by adding a formerly generated data address and a stride value generated from the difference between two consecutive generations of the data address, the prediction

address advantageously predicts the next address correctly for at least two types of data access patterns. First, if an instruction accesses the same address each time it executes, then the stride value will be zero and the data prediction address associated with that instruction will be the same each time the instruction executes. Therefore, the current data prediction structure will correctly predict static data addresses. Additionally, instructions which access a regular pattern of data addresses such that the data address accessed on consecutive executions of the instruction differ by a fixed amount may receive correct address predictions from the present data prediction structure. Static data prediction structures (which store the previously generated data address without a stride value) do not predict these types of instruction sequences correctly. Therefore, the present prediction structure may correctly predict a larger percentage of data addresses than static data prediction structures.

Turning now to Figure 3B, an embodiment of reservation station 210A is depicted. One storage entry is shown in Figure 3B, but reservation station 210A is configured with several storage entries for storing decoded instructions and related data. Other storage entries within reservation station 210A are configured similar to the storage entry shown. Additionally, reservation stations 210B-210F are configured similar to reservation station 210A. The logic circuits shown in Figure 3B are exemplary circuits for implementing the data prediction structure. Other circuits (not shown) select instructions for execution by functional units 212.

The reservation station storage entry shown in Figure 3B includes storage for the decoded instruction and related information. A decoded instruction register 300 stores a decoded instruction conveyed from decode unit 208A on an input instruction bus 303. The input operands are stored in a pair of registers: AOP register 301 and BOP register 302. AOP register 301 and BOP register 302 receive their respective values either from a pair of data forwarding buses 304 and 305 from functional units 212 through operand steering logic (not shown), or from a predicted data register 306. The selection of data to store within AOP register 301 and BOP register 302 is performed by multiplexors 307 and 308, respectively, under the control of a reservation station control unit 309. AOP register 301 and BOP register 302 are each configured with a valid bit indicative of whether or not a value has been provided for the corresponding operand. When both operands are valid, the explicit operation of the associated instruction may be executed by functional unit 212A.

Predicted data register 306 is coupled to second reply bus 259. During a clock cycle in which a decoded instruction is first stored into decoded instruction register 300, data from second reply bus 259 is stored into predicted data register 306, along with an indication of whether or not the address associated with the data is the data prediction address and whether or not the address was a hit in data cache 224. During this clock cycle, the address used to fetch the instruction from instruction cache 204 is stored into instruction address register 310. Additionally, the associated data prediction address and data prediction counter value generated by branch prediction unit 220 are stored into a predicted address register 311. Predicted address register 311 is configured with a validation done bit (represented in Figure 3B by the D field of predicted address register 311). The validation done bit is used to ensure that a particular predicted address is

validated only once by reservation station 210A and conveyed to branch prediction unit 220. The validation done bit is initially cleared by reservation station control unit 309 and subsequently set as described below. Finally, the storage entry includes a linear address register 312 for storing a linear address generated for the decoded instruction by functional unit 212A. Additionally, a valid bit is stored in linear address register 312 to indicate the validity of the linear address.

Reservation station control unit 309 is configured to control the storing of data into the various registers within each reservation station storage location. A decoded instruction is conveyed to the storage location and stored in decoded instruction register 300. During the same clock cycle, predicted data register 306 receives data from data cache 224 and instruction address register 310 and predicted address register 311 receive values routed through decode unit 208A from branch prediction unit 220. Additionally, AOP register 301 and BOP register 302 may receive operand values from data forwarding buses 304 and 305. The decoded instruction occupies the storage entry within reservation station 210A until the explicit operation of the instruction is executed by functional unit 212A.

If the decoded instruction utilizes an implicit memory read operation to retrieve one of its operands, reservation station control unit 309 causes functional unit 212A to generate a linear address for the implicit memory read from the appropriate address values (not shown). The linear address is conveyed to load/store unit 222 for use in performing the implicit memory read if the data prediction address is incorrect, and is additionally stored into linear address register 312. Reservation station control unit 309 causes the valid bit in linear address register 312 to be set. During a subsequent clock cycle, a comparator circuit 313 compares the generated linear address to the data prediction address stored in predicted address register 311. Comparator circuit 313 is configured to indicate a mismatch if the data prediction counter value stored in predicted address register 311 indicates that the data prediction address is invalid (i.e. the associated storage location within prediction array 270 is not currently storing a valid base address).

Comparator circuit 313 conveys its output signal indicative of a match or mismatch between the data prediction address and the linear address to a multiplexor 314. Additionally, the value stored in instruction address register 310 is conveyed to a multiplexor 315; the value stored in predicted address register 311 is conveyed to a multiplexor 316; and the value stored in linear address register 312 is conveyed to a multiplexor 317. Multiplexors 314, 315, 316, and 317 are controlled by reservation station control unit 309 and receive similar values from other reservation station storage entries (not shown). During a clock cycle in which the valid bit within linear address register 312 is set and the validation done bit within predicted address register 311 is clear, reservation station control unit 309 selects the values stored in the associated storage entry through multiplexors 314, 315, 316, and 317 and sets the validation done bit in predicted address register 311. Multiplexor 314 is configured to convey a value on mispredicted line 273A, which is one of the plurality of mispredicted lines 273 shown in Figure 3B. Similarly, multiplexor 315 is configured to convey a value on instruction address bus 274A, which is one of the plurality of instruction address buses 274 shown in Figure 3B. Multiplexor 316 is configured to convey the value stored in predicted address register 311 on prediction address bus 276A, which is one of plurality of prediction

address buses 276. Finally, multiplexor 317 is configured to convey the value stored in linear address register 312 (including the valid bit, which forms the address valid indicator described above) on linear address bus 261A, which is one of plurality of linear address buses 261. In this manner, the values used by prediction validation and correction block 275 of branch prediction unit 220 (shown in Figure 3A) receive the appropriate values to validate and correct the data address prediction generated for the instruction stored in decoded instruction register 300.

During the clock cycle that the values associated with the instruction in decoded instruction register 300 are conveyed to branch prediction unit 220, reservation station control unit 309 sets the validation done bit in predicted address register 311. Since the validation done bit must be clear for validation values to be conveyed by the reservation station storage entry, the validation in branch prediction unit 220 is performed once for a particular instruction. If the linear address and data prediction address are found to match and the data stored in predicted data register 306 is associated with the data prediction address, reservation station control unit 309 causes the data stored in predicted data register 306 to be stored into either AOP register 301 or BOP register 302. AOP register 301 or BOP register 302 are chosen according to which operand is intended to receive the result of the implicit memory read operation. If the linear address and data prediction address do not match, the implicit memory read operation is performed by load/store unit 222 in a subsequent clock cycle. If the predicted data is transferred to AOP register 301 or BOP register 302, then reservation station control unit 309 conveys an indication to load/store unit 222 that the associated implicit memory read operation need not be performed. Load/store unit 222 thereby discards the implicit memory read operation (as noted above).

If none of the reservation station storage entries within reservation station 210A are in condition to convey prediction validation information during a clock cycle, reservation station control unit 309 causes the address valid indicator on linear address bus 261A to convey an invalid address indication. Therefore, branch prediction unit 220 will ignore the values conveyed on linear address bus 261A, prediction address bus 276A, instruction address bus 274A, and mispredicted line 273A during this clock cycle. Reservation stations 210B-210F thereby may update data prediction values during this clock cycle.

Turning now to Figure 4, a diagram of the fields within a storage location 400 of prediction array 270 is shown. A field of information stored in storage location 400 of prediction array 270 is the base address field 401. The base address is stored in this field. In one embodiment, base address field 401 is 32 bits wide. Associated with the base address is a stride value stored in stride value field 402. In one embodiment stride value field 401 is 10 bits wide. The base address and stride value are added to form the data prediction address.

Finally, data prediction counter field 403 is included in storage location 400. The data prediction counter is stored in this field. In one embodiment, the data prediction counter is two bits. A binary value of 00 for the data prediction counter indicates that neither the base address field 401 nor the stride value field 402 contain valid values. A binary value of 01 indicates that base address field 401 is valid but stride value

field 402 does not contain a valid value. The base address field is added to zero for this encoding, and the stride value is updated when the linear address is generated by one of decode units 208. A binary value of 10 or 11 indicates that both base address field 401 and stride value field 402 are valid. Having two values with the same indication allows for a stride value to be maintained until two consecutive mispredictions are detected for basic blocks which index storage location 400. In this way, prediction accuracy is maintained for the case where a basic block is intermittently executed between multiple executions of a second basic block and the two basic blocks index the same storage location 400. Other embodiments may implement more or less bits for the data prediction counter value.

In another embodiment of the data address prediction structure, data prediction address bus 253 is connected to an additional read port on data cache 224. In this embodiment, arbitration multiplexor 254 may be eliminated. It is noted that, although the preceding discussion described an embodiment of the data prediction structure within a processor implementing the x86 architecture, any microprocessor architecture could benefit from the data prediction structure.

In accordance with the above disclosure, a data address prediction structure is described which allows an implicit memory operation of an instruction to be performed before the instruction reaches a reservation station. Therefore, the number of cycles that the instruction is stored in the reservation station is advantageously reduced. Performance may be increased by implementing such a data prediction structure.

Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

WHAT IS CLAIMED IS:

1. A method for predicting a data address which will be referenced by a plurality of instructions when said plurality of instructions are fetched, comprising:

generating a data prediction address from a base address and a stride value during a clock cycle in which a data prediction counter indicates that said base address and said stride value are valid;

fetching data associated with said data prediction address from a data cache; and

storing said data within a plurality of reservation stations.

2. The method as recited in claim 1 wherein said base address, said stride value, and said data prediction counter are stored in a prediction array.

3. The method as recited in claim 2 further comprising updating said prediction array with a new base address during a second clock cycle in which a second data address is generated by executing said plurality of instructions.

4. The method as recited in claim 3 further comprising incrementing said data prediction counter during said second clock cycle if said data prediction address is correctly predicted.

5. The method as recited in claim 4 further comprising updating said prediction array with a new stride value during a second clock cycle if said data prediction counter indicates that said stride value is invalid.

6. The method as recited in claim 5 wherein said new stride value is a difference between said data prediction address and said second data address.

7. The method as recited in claim 5 further comprising decrementing said data prediction counter during said second clock cycle if which said data prediction address is found to be mispredicted and said data prediction counter indicates that said stride value is valid.

8. The method as recited in claim 1 further comprising generating an actual data address by executing said plurality of instructions.

9. The method as recited in claim 8 further comprising comparing said actual data address to said data prediction address.

10. The method as recited in claim 9 further comprising transferring said actual data address and a result of said comparing to a branch prediction unit.

11. The method as recited in claim 1 wherein said generating a data prediction address is performed during a third clock cycle in which said plurality of instructions is being fetched from an instruction cache.

12. The method as recited in claim 1 further comprising generating said data prediction address from said base address during a fourth clock cycle in which said data prediction counter indicates that said base address is valid but said stride value is invalid.

13. A data address prediction structure comprising:

an array including a plurality of storage locations for storing a plurality of base addresses and a plurality of stride values wherein a particular one of said plurality of storage locations in which a particular one of said plurality of base addresses and a particular one of said plurality of stride values are stored is selected by an instruction address, and wherein said instruction address identifies an instruction which generates said particular one of said plurality of base addresses;

an adder circuit coupled to said array for adding a base address and a stride value conveyed from said array to produce a data prediction address; and

a reservation station for storing said instruction and data associated with said data prediction address.

14. The data address prediction structure as recited in claim 13 wherein said array is a linear array of said storage locations.

15. The data address prediction structure as recited in claim 13 further comprising a data cache coupled to said adder circuit, wherein said data prediction address is used to fetch data from said data cache, and wherein said data is conveyed to said reservation station.

16. The data address prediction structure as recited in claim 13 wherein said reservation station further comprises a comparator circuit, a first register, and a second register wherein said comparator circuit is coupled to said first register and to said second register.

17. The data address prediction structure as recited in claim 16 wherein said first register is configured to store said data prediction address.

18. The data address prediction structure as recited in claim 16 wherein said second register is configured to store a linear address computed by executing said instruction.

19. The data address prediction structure as recited in claim 16 wherein said comparator circuit is configured to produce a value indicative of the correctness of said data prediction address.

20. The data address prediction structure as recited in claim 19 wherein said comparator circuit is configured to convey said value to said array.

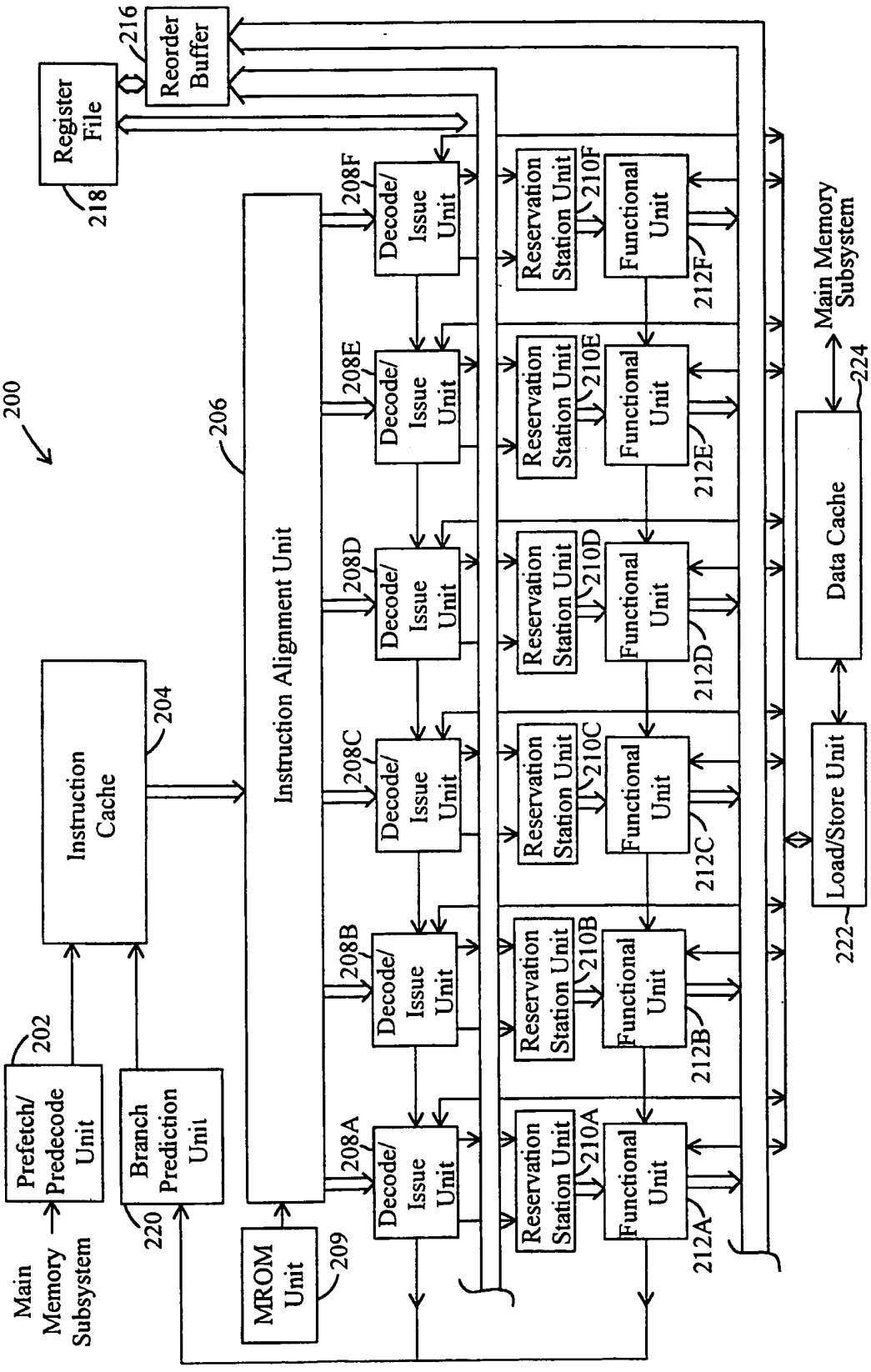

200
Main Memory Subsystem
Prefetch/ Predecode Unit
202
Branch Prediction Unit
220
Instruction Cache
204
MROM Unit
209
Instruction Alignment Unit
206
Register File
218
Reorder Buffer
216
Decode/ Issue Unit
208A
208B
208C
208D
208E
208F
Reservation Station Unit
210A
210B
210C
210D
210E
210F
Functional Unit
212A
212B
212C
212D
212E
212F
Load/Store Unit
222
Data Cache
224
Main Memory Subsystem

Fig. 1

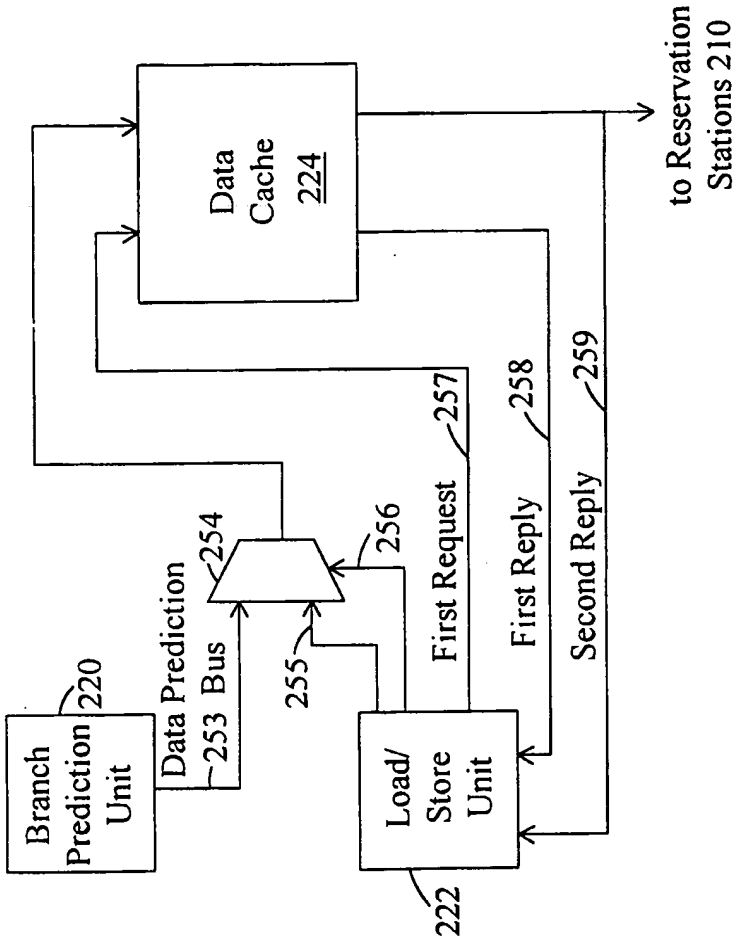

Data Cache 224
to Reservation Stations 210
Branch Prediction Unit
220
Data Prediction 253 Bus
254
256
255
First Request
257
First Reply
258
Second Reply
259
Load/ Store Unit
222

Fig. 2

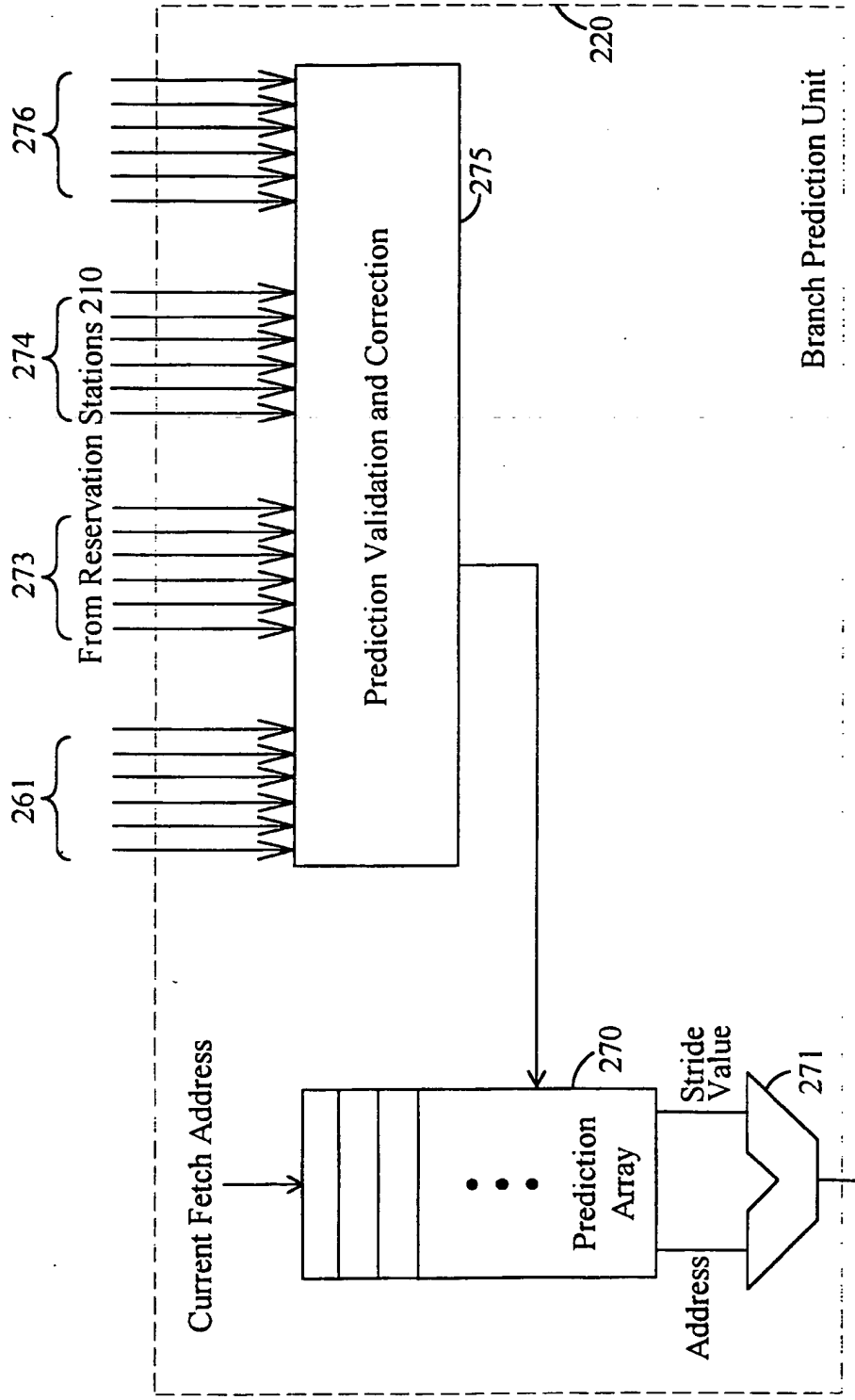

220
Branch Prediction Unit
276
274
273
261
From Reservation Stations 210
275
Prediction Validation and Correction
270
Prediction Array
Stride Value
271
Address
Current Fetch Address
Data Prediction Bus 253

Fig. 3A

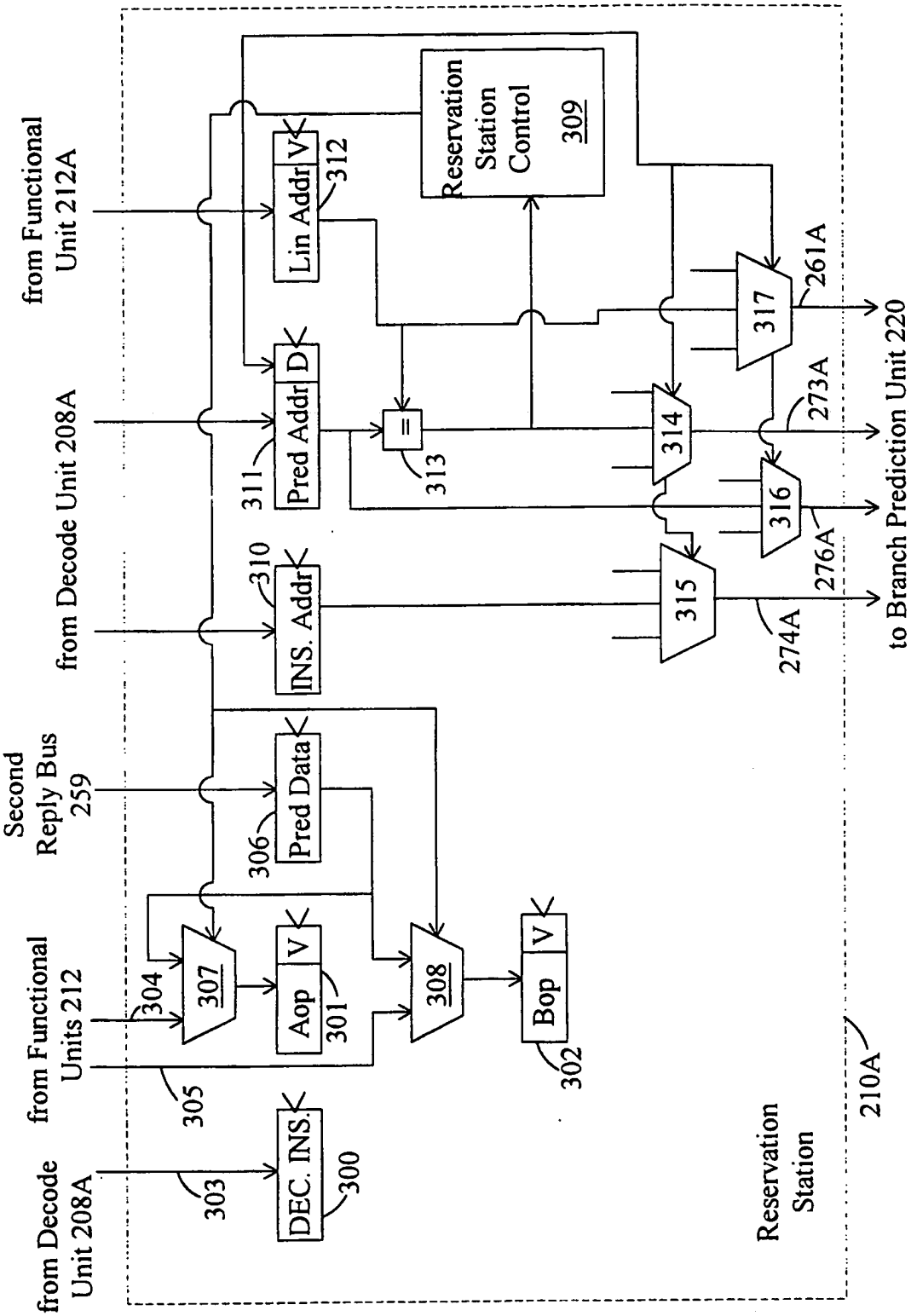

from Functional Unit 212A
from Decode Unit 208A
Second Reply Bus 259
from Functional Units 212
from Decode Unit 208A
Reservation Station
210A
Reservation Station Control 309
Lin Addr V 312
Pred Addr D 311
INS. Addr 310
Pred Data 306
Aop V 301
Bop V 302
DEC. INS. 300
304
305
303
307
308
=
313
314
315
316
317
261A
273A
276A
274A
to Branch Prediction Unit 220

Fig. 3B

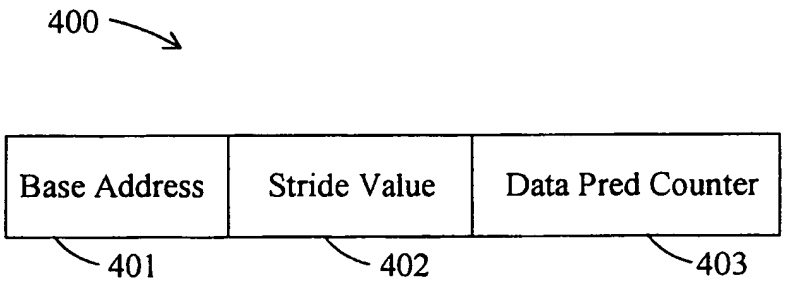


Fig. 4

INTERNATIONAL SEARCH REPORT

Internatio pplication No
PCT/US 96/17516

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/38

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 5 442 767 A (EICKEMEYER RICHARD J ET AL) 15 August 1995 see the whole document ---	1-3,8-20 4-7
X	PROCEEDINGS OF THE SUPERCOMPUTING CONFERENCE, ALBUQUERQUE, NOV. 18 - 22, 1991, no. CONF. 4, 18 November 1991, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 176-186, XP000337480 BAER J L ET AL: "AN EFFECTIVE ON-CHIP PRELOADING SCHEME TO REDUCE DATA ACCESS PENALTY" see the whole document --- -/--	1,13

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
10 July 1997	22.07.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Daskalakis, T

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Internatic .pplication No
PCT/US 96/17516

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	PROCEEDINGS OF THE 28TH. ANNUAL INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE, ANN ARBOR, NOV. 29 - DEC. 1, 1995, no. SYMP. 28, 29 November 1995, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 252-257, XP000585367 PO-YUNG CHANG ET AL: "ALTERNATIVE IMPLEMENTATIONS OF HYBRD BRANCH PREDICTORS" see page 253, right-hand column ---	4-7
A	IBM JOURNAL OF RESEARCH AND DEVELOPMENT, vol. 37, no. 4, July 1993, pages 547-564, XP000647102 EICKEMEYER R J ET AL: "A LOAD-INSTRUCTION UNIT FOR PIPELINED PROCESSORS" see the whole document -----	1-20

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Internat Application No
PCT/US 96/17516

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5442767 A	15-08-95	NONE	